intel®

APPLICATION NOTE

AP-246

October 1985

Setting Up an Efficient Hierarchical File System

WAYNE ROSEN
DSO APPLICATIONS ENGINEERING

Order Number: 231482-001

INTRODUCTION

Software development has become a team activity.

— Team members need an efficient file management scheme.

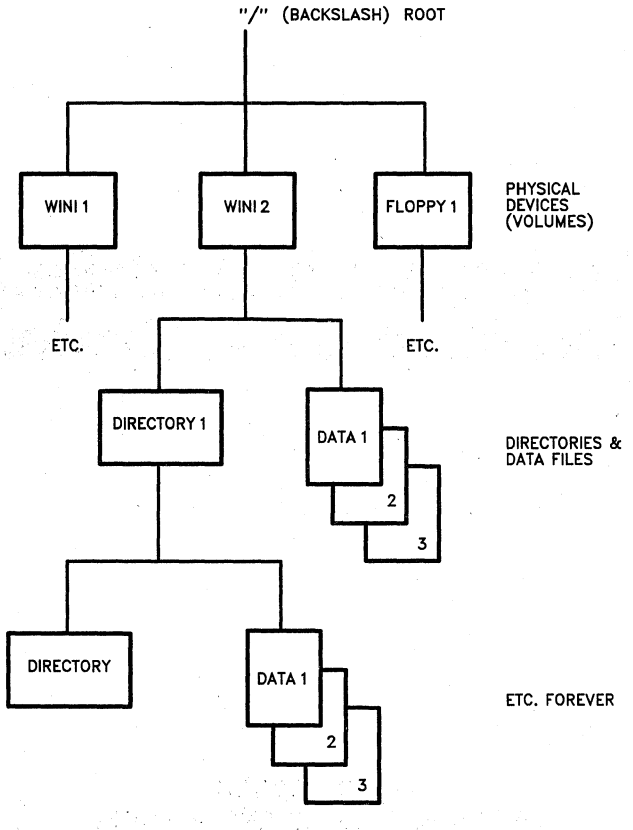
— Team members need to share common databases, but need to be protected against unauthorized file access.

Intel provides a superior hierarchical file system for file management, protection, and sharing in a totally controlled environment.

This Application Note is directed to the NDS-II or Series IV SUPERUSER who is setting up the system's hierarchical file system (HFS) and software development environment. We will be using some hypothetical products to illustrate our recommendations for this HFS.

Intel's NDS-II Network Resource Manager (NRM) and Series IV, running the iNDX operating system, encourage a logically constructed HFS. However, unless set up in a well-structured manner, an HFS can cause many problems. As software tasks grow larger and more complex, a properly structured file system will speed overall system development.

An HFS, (a tree-type file system opposed to a flat file system), promotes system protection and project partitioning and allows users to quickly find needed files. Figure 1 shows a stylized HFS.



NOTE:
1. A "volume" is a mountable, physical device. The system maintains device names for these, which should not be confused with the names you give them. For example:
Device name WFO is the 85 MB Winchester drive controlled by the disk controller board's first driver circuit.
Device name WF1 is the 84 MB Winchester drive controlled by the disk controller board's second driver circuit.

Figure 1. A Stylized HFS

ADVANTAGES OF AN HFS

The following are properties of Intel's HFS (See Figure 1).

- All files have a unique pathname starting from the root.
- Each physical device represents a directory at the root.
- Directories may contain data files or more directories.

Where the root (represented by "/") is the symbolic connection point for all physical volumes of an HFS.

To determine the physical volumes available to you as a user, enter the command "DIR /". For example:

```
DIR /
iNDX-W41 (V2.8) DIR V2.8
DIRECTORY OF /

FILE_NAME LOCATION  ACCESSIBILITY

SYS         remote    ;an NDS-II device
APS0        remote    ;     ""
APS1        remote    ;     ""
WLR.BACKUP  local     ;a local device on
                       my Series IV
```

A Logical Place to Put "Things"

A big advantage to using an HFS is the ability to group files according to user-defined relationships. Let's illustrate this important feature with a story.

We live in a disorganized universe. The laws of entropy tend to maintain and promote this disorganized state. Human beings fight the forces of entropy and try to maintain order in the small niche they carved out for themselves.

In our small corner of the universe, some people like "a place for everything and everything in its place;" that is, they expend some energy organizing their life and surroundings, while others do not bother. Joe Slobotnick (a very bad NDS-II manager) leans towards maximum entropy. Joe does not bother to expend the minimal energy necessary to maintain order on his system.

Joe is the only one in the world who might know where something is kept on the system. Occasionally, even Joe forgets where something needed is stored ("I swear I put that file in the TEMP3 directory along with the other prototypes") At this point, a mad, random-access search begins. This frantic search follows no known rules (like a binary or Shell sort), and no maximum search time can be calculated to tell Joe how long the search will take. Thus, the frantic search may take considerable time, may not be worth the effort, and may have disastrous side effects (like never finding the object of the search).

How much simpler it is for Joe, and for anyone working on his system, if a logical structure is imposed on the system. More importantly, how much simpler for all if a minimal effort is exerted to maintain this logical order.

Protection

Another advantage to using INTEL's HFS properly is file protection. iNDX provides the capability to protect critical files not only from malicious tampering, but from accidental changes (accidents do happen!). For example, all users (even SUPERUSER) should be able to use a compiler; but they should not be able to change or delete it.

Every file in INTEL's HFS has an "owner" associated with it. This owner is someone the SUPERUSER has defined as a system user (see the USERDEF utility). This owner controls access rights to his or her files by:

- Setting the individual access rights
- Setting the world's (the rest of the users on the system) access rights.

Superuser (including those people with secondary Superuser rights) can override the built-in protections and do anything to your files. This is a good reason to restrict the use of Superuser authority to the absolute minimum.

The Software Version Control System (SVCS), Intel's database manager, maintains another level of protection over that provided by the HFS. Features of this utility are discussed in Application Note AP 162 - a PMT tutorial.

Your Home Directory

You will want to keep your personal files in a protected directory that you own. This directory should be your home directory defined at USERDEF time.

When you log on, the iNDX operating system will automatically assign the logical names ' ' (the NULL logical name) and WORK: to your home directory.

```
LNAME Path
iNDX-W41 (V2.8) LNAME V2.8
LOGICAL      NAME PATHNAME
" "          /volume/USER.DIR/WAYNE.DIR
:WORK:       /volume/USER.DIR/WAYNE.DIR
```

Utilities will default their operations to the NULL logical name if no directory is specified, that is, the DIR command will give a directory listing of my home directory. PLM86 SOMEFILE.PLM will look for the file SOMEFILE.PLM in my home directory. In addition, the NULL logical name is the starting point to easily reference subdirectories located in your home directory. For example:

```
DIR MEMOS.DIR                ;MEMOS.DIR is
a sub-directory
iNDX-W41 (V2.8) DIR V2.8  ;in my home
directory
DIRECTORY OF /volume/USER.DIR/
 WAYNE.DIR/MEMOS.DIR -full pathname
FILE_NAME    FILE_NAME    FILE_NAME
MANPOW.D14   MAILD.323    UPGRAD.D12
CONF.305     SUNEWS.127   HFS.612
VACATION.N14
```

The NULL logical name can be redefined, but we do not recommend it.

The :WORK: logical name is used by various utilities (including translators) for workspace. We do recommend redefining this logical name. For example, if you are logged onto a network and have an Series IV with a Winchester (a fast device), defining :WORK: to some working directory on your local Winchester will speed you own processing and will reduce overall network Ethernet traffic. In fact, any temporary files created by you should be dispatched to this :WORK: directory. At the end of the day, you can clean up your workspace by simply deleting this working directory.

As you accumulate additional files in your home directory, you should break off related files into subdirectories, such as:

```
MEMOS.DIR          ;all memos
   KEYRESULTS.DIR;those memos that are
key results
```

In fact, we recommend that, other than needed initialization or configuration files, only put other directories in your home directory. Figure 2 is a sample home directory.

Under iNDX, the maximum directory name is 14 alphanumerics. Periods as readability delimeters count as one of the 14 characters.

Under ISIS-III(N), the maximum directory name is 6 dot 3. That is, six alphanumerics, a period and three alphanumerics for the optional extension. Also, it is convenient to name memos in the following form:

```
name.date_code
```

Where date_code = Mdd (three alphanumerics)

M = month code
(1-9 for Jan through Sep, 0 for Oct, N for Nov, D for Dec)
dd = day of the month (01 - 31)

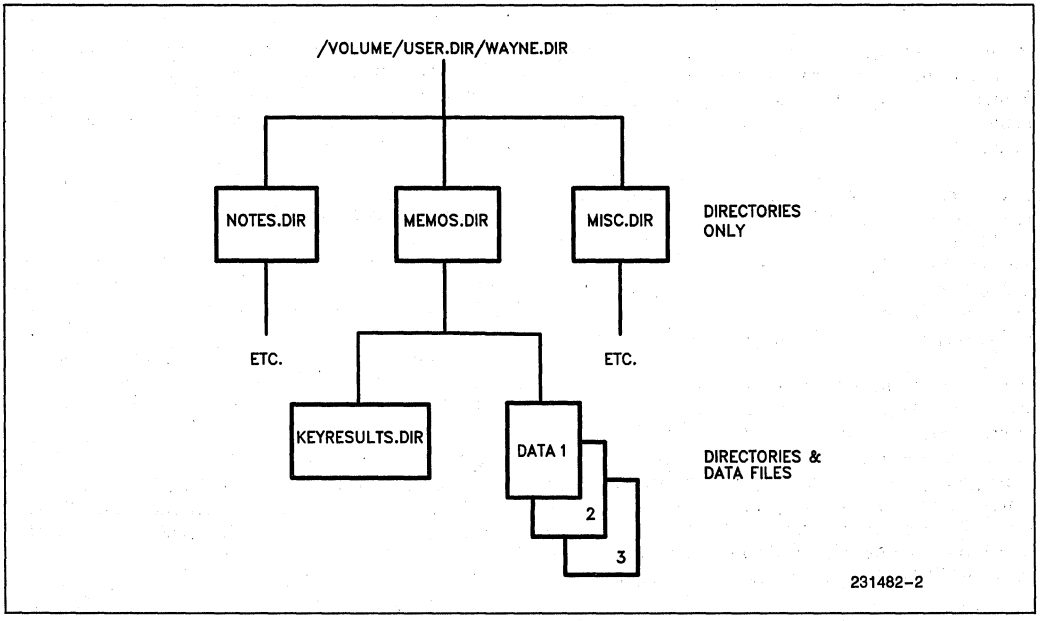


Figure 2. A Sample Home Directory

One Place For Tools

Tools (compilers, linkers, editors, etc.) should be kept in only one location. The world and owner should have DISPLAY ACCESS rights only.

This centralized tool directory is very convenient. Since there is only one copy of each of the tools, the system manager can guarantee that:

- Everyone is using the same version of the tools
- Tool updates need be made in one place only
- When a system generation is done, it can be proven that every module was generated using the same tools.

This last point is very important when it comes to system validation and certification. The Department of Defense (DOD), the Federal Aviation Administration (FAA), and others require such version control guarantees.

What takes place inside the computer when a command is invoked? For example:

```
PLM86 some.file Debug
```

Based on user and system-defined search rules (discussed later in this note), the operating system will begin searching specific directories for the PL/M86 compiler. If the compiler is not found in the first directory, the operating system will then search subsequent directories.

How does the operating system know if a file is in a directory? The operating system performs a linear search through the file entries until a match is found. Files marked as "deleted" and subdirectories count as entries too. The average "match time" is:

(½) × (total # of file entries) × (time to perform the match function)

To optimize system performance, you will want to:

- Make certain that the most frequently used utilities are located in the first directory searched
- Order the utilities in this first directory to minimize "match time" for very frequently used utilities,, such as AEDIT, DIR, and COPY (put them into the tools directory first)
- Minimize the total number of files in a directory (especially the tools directory). The maximum number of files that iNDX will allow in a directory is 1,024.

The Problem With Dir

Doing a directory (DIR) listing has to be one of the most frequently used commands of any computer system. However, doing a DIR on an unsorted directory that contains 756 files not only takes a lot of time, but limits the probability of locating all desired files. Your brain and eyes have a difficult time scanning pages of scrolling directory listings. Approximately 75 files (one page of a three-column DIR listing) is the recommended maximum amount of files to be scanned at one time.

How do you pick out subdirectories in a DIR listing? Unless you know the names of those subdirectories, an expanded DIR is the only way to find those directories. Expanded DIRs take a long time and degrade overall network performance. The answer? Where possible, suffix all directories with .DIR. Under iNDX, you have up to 14 characters to specify a directory name. This way, you will be able to find your subdirectories with a standard DIR listing. Or, you can search for occurrences of .DIR only. For example:

```
DIR directory_name FOR *.DIR
```

However, you might prefer having 756 files (or more) in your directories. When you look for a particular file(s), you will use wildcard characters and match for a particular pattern. Unfortunately, this also takes time. Then, there is the problem of possibly missing a needed file that does not quite match the search pattern (or getting other extraneous files that do match the pattern).

The bottom line is this: If you have a system that supports an HFS, use it wisely! And, be sure to:

GROUP RELATED FILES UNDER A
M E A N I N G F U L L Y
NAMED DIRECTORY !!

A SAMPLE PROJECT

The following sample project will help illustrate how to set up an NDS-II hierarchical file system. There are many projects being developed on our NDS-II network. The one project our group is working on is:

ROBOTWELDER a dual 186-based project

Our development environment has the following components:

- One NDS-II
- built-in tape cartridge (for back-up)
  - — One 35 MB Winchester (what we originally ordered)
  - — One 84 MB Winchester (we bought this unit when we needed more disk space)
- Two Series IVs
  - — One flippy/winny
  - — One flippy/flippy

- Two Series IIs (our original boxes, pre-network)
- One Series III
- Four ISIS cluster stations
- Intel in-circuit emulators as needed.

A MODEL HFS

Software is divided into three worlds:

*** program equivalent ***

TOOLS: editors, compilers, linkers, etc. (code)

USERS: you, me, and our projects (data)

SYSTEM: network operations (operating system)

In general, the tools operate on output from the users,

Users' files = Tools (users' files)

The system software is responsible for the operation of the computer. The system software manages the tools, the users' files, and itself.

Network operation = System (Tools, users' files, system)

Under iNDX, the system software is responsible for file protection, distributed job control, resource sharing, electronic mail, etc.

Using The Winchesters

Since we have this particular Winchester configuration (see "Sample Project"), we will use the 35 MB Winchester as the boot and system device. In fact, we will make this disk "read only". All of our tools (which have read permission only) will reside on this disk. We get a performance benefit by making this disk read only. A disk write takes approximately seven times longer than a disk read (reduced head thrashing). In our particular configuration, we gain added performance, since there are separate disk controllers for the 35 MB and 84 MB Winchesters (each type of controller can support up to four disks).

Since this 35 MB disk has our tools, contains our system software, and is the boot disk, we will name it something meaningful. For example:

```
/SYS
```

not simply /W or /WO.

Later, we will show why limiting this name to three alphanumerics is useful. An early hint:14 characters is the maximum that the SEARCH CUSP presently accepts.

* Wordstar is a trademark of Micropro.

** Multiplan is a registered trademark of Microsoft Corp.

You should always have at least 2 MB of spare room on the boot disk. iNDX creates many temporary files, some quite large, and puts them on this boot disk. For example, the SPOOL directory is the temporary holding space for print jobs. If you have sent 500k worth or listings (all at once) to be printed, you will need at least 500k free on the boot disk.

Tools

Let us take an in-depth look at these software tools. As far as our NDS-II and workstations are concerned, these tools are divided between:

8-bit tools

— These tools run on the 8085 microprocessor.

— The Series II, cluster board, and Model-800 can run ONLY these tools.

— The hosted 8-bit operating systems are ISIS and CP/M.

16-bit tools

— These tools run on the 8088/8086 microprocessors.

— The Series III and Series IV run these tools (in addition to being able to run all the 8 bit tools).

— The hosted 16-bit operating systems are iNDX and ISIS RUN.

As far as CP/M is concerned, Intel's development tools do not run under CP/M. CP/M is useful if you wish to include others into the development process for example, the professional using Wordstar* and the financial planner using Multiplan.** CP/M running on a workstation on the network is discussed in depth DSO Application Note AP-253:Adding Value to Intel's NDS-11 Development System Network with Network CP/M-80.

It is a misconception to believe that tools running on a 8-bit machine can only generate objects that an 8-bit microprocessor can use. Intel supplies a 8-bit PL/M compiler (i.e., runs under ISIS) that generates object code for an 8088/8086 microprocessor.

Due to the inertia of history or tradition, the 8-bit world is called:

```
ISIS.SYS (it would have been nice to
call it 8BIT.DIR).
```

However, we can call our 16-bit world:

```
16BIT.DIR.
```

16BIT.DIR

It is useful and very convenient to subdivide ISIS.SYS and 16BIT.DIR into logical groups. Under ISIS, we have great flexibility to do this. A Series IV can specify one additional Search path right now.

We would like to digress a bit and talk about the iNDX SEARCH CUSP. Under ISIS, when a CUSP is invoked or referenced by just its name, the command line interpreter (CLI) looks for the CUSP in the default directory, :FO:. For example:

```
  DIR (this is the same as typing
:FO:DIR)
```

Similarly, under iNDX, when a CUSP is invoked or referenced by just its name, the CLI:

- First looks in the system volume directory (in our example, this is called /SYS).
- If not found, the CLI then looks in the directory specified by the NULL logical name (''). The NULL Logical name is defaulted to your home directory and should not be changed.

It is convenient to have additional search paths. The current SEARCH CUSP gives us one more, which we can use to point to 16BIT.DIR:

```
SEARCH /SYS/16BIT.DIR
```

Thus, for our Series IVs, our search paths are:

```
 /SYS/16BIT.DIR
 /SYS                              ;boot
device
 /WORK/USER.DIR/home_directory     ;the
NULL logical name
```

The CLI will first search /SYS/16BIT.DIR, then the boot device, and finally our home directory. A CUSP found in any of our search directories with an entry in the menu compiler, will be able to:

— Access its syntax builder

— Complete command lines (FILL ON)

— Display HELP messages for any portion of the command line.

Why did we limit the system volume name to three alphanumerics? Currently, the search pathname, /SYS/16BIT.DIR, cannot be longer than 14 alphanumerics (including backslash delimeters). Our way to get around this limitation (if you have a longer volume name) is to use an LNAME.

```
  LNAME define 16BIT.DIR for
/long_volume_name/16BIT.DIR,
```

But then:

— You use one more LNAME

— This LNAME cannot be removed or redefined

— All users have to set up this LNAME.

ISIS.SYS

It seems that everyone sets up his or her own virtual floppy assignments in individual ISIS.INI files. We recommend that the group adopt a common standard and stick with it. We suggest the following:

```
                          *** 8-bit workstation ***
ASSIGN     :F0:to /SYS/ISIS.SYS
ASSIGN     :F1:to :F9:today's_project.DIR (your working directory)
ASSIGN     :F2:to /work_volume/PROJECT.DIR/xxx.DIR/DATABASE.DIR/MODULE.dir
     ;xxx is the project you're working on
     ;module is the particular piece of the project you're working on at the
      moment (if a large project)
ASSIGN     :F3:to /SYS/ISIS.SYS/LIB.DIR
     ;libraries, system $INCLUDE files
ASSIGN     :F4:to /SYS/ICE.DIR/which_ICE_you're_using.DIR
     .
     .
     .
ASSIGN :F9:to /work_volume/USER.DIR/home_directory.DIR
```

NEVER, NEVER re-ASSIGN :F9:, your home directory.

16-BIT WORKSTATION (SERIES III)

This is a Series III in RUN mode. In addition to the assignments above, add the following:

```
ASSIGN    :F7:to /SYS/16BIT.DIR/LIB.DIR      ;16-bit libraries
ASSIGN    :F8:to /SYS/16BIT.DIR              ;16-bit CUSPS
```

DIRECTORY STRUCTURE FOR TOOLS

We will put all interactive software tools under one of the following directories:

```
/SYS/ISIS.SYS, or
/SYS/16BIT.DIR
```

based on whether the tools are hosted on an 8-bit or a 16-bit processor.

Since commonly used $INCLUDE files (.H files for you "C" people) and libraries are nonexecutable and are usually brought in during a SUBMIT file, we can put them into subdirectories:

```
/SYS/ISIS.SYS/LIB.DIR, and
/SYS/16BIT.DIR/LIB.DIR
```

Project-related $INCLUDE files should be stored in the project database.

There are other files that have nothing to do with the host processor, such as configuration files that set up a terminal for an editor or PSCOPE or configure a terminal for a second user (Series IV). These we will put under:

```
/SYS/CONFIG.DIR
```

Our target processor (the processor(s) in our product) has nothing to do with the host processor that our development system is running. For this reason and for modularity and partitioning purposes, we have elected to break out all of the ICE emulator software and lump it under a separate directory:

```
/SYS/ICE.DIR
```

For convenience, certain CUSPs should be kept in the root directory of the boot device. We suggest that all you need to leave behind are:

```
LOGON    ;Never remove from the root
(see Appendix A)
DIR.86
LOGOFF.86
```

You may be wondering why we chose to remove as many CUSPs out of the root as possible. The root directory of the boot device is already cluttered with many system files; the total number can be substantial. (See Appendix 1 for a list of these system files.)

Figure 3 contains our suggested directory structure for tools.

```
/SYS                                    ;volume name
        /16BIT.DIR                      ;16-bit tools
                /LIB.DIR                ;libraries
        /ISIS.SYS                       ;8-bit tools
                /LIB.DIR                ; libraries
        /CONFIG.DIR                     ;configuration files
                                        ; for various terminals
                /AEDIT.MAC.DIR          ; AEDIT
                /STTY/CFG.DIR           ; Series IV users
        /ICE.DIR                        ;in-circuit emulators
                /ICE51.DIR
                /ICE.86.DIR
                /I2ICE.DIR
                    .
                    .
                    .
```

Figure 3. Directory Structure for Tools.

The first thing your INIT.CSD (Series IV LOGON initialization file) should do is:

```
SEARCH /SYS/16BIT.DIR
```

THE CASE OF THE MISSING CUSPS

You have looked everywhere on your disk, but you simply cannot find the EXPORT.86 file. Has someone deleted it? No! This iNDX CUSP, and other "hidden" CUSPs listed below, are built directly into the iNDX CLI.

***COMMAND FILE PROCESSING

BATCH	command file editor with syntax help
SUBMIT	executes the command file instantaneously
EXPORT	executes the command file at another time and place
BACKGROUND	executes the command file in my background now

***COMMAND FILE CONTROLS

COUNT	allows multiple executions of commands
REPEAT	allows multiple executions of commands
UNTIL	used by COUNT and REPEAT
WHILE	used by COUNT and REPEAT
ENDJOB	terminates this command file now
OPEN	opens a parameter file
READ	gets a parameter from a file
SET	allows (re-)definition of CLI variables
IF	conditional execution
ORIF	conditional execution
ELSE	conditional execution

***MISCELLANEOUS CUSPS

SEARCH	enables or lists CLI search paths
FILL	enables disables CLI command completion
LOG	saves all console output to a file
END	noop command for ISIS compatability
RUN	noop command for ISIS compatability
VIEW	scan a file (AEDIT-like interface)

The command file controls (IF, UNTIL, etc) are very useful for controlling command file (SUBMIT) execution. (Refer to the Application Note AP 245 for further discussion.)

Users

The next part of our software world is for the users. The first part of this world contains all our personal, home directories. The second part of this world contains all the project files.

The following example shows setting up this directory. We are using our 84 MB Winchester as our work disk and, therefore, are naming it WORK1.

PEOPLE

```
/WORK1                                  ;volume name
     /USER.DIR                          ;main directory
               /SUPERUSER.DIR           ;with release 2.8 of iNDX, the
                                         superuser
                                        ;gets his or her own home directory
               /WAYNE.DIR
                    /MEMO.DIR                ;as an example
                         /KEYRESULTS.DIR      as an example
                         /APNOTES.DIR         ;as an example
                    /BRIAN.DIR
                    /CHRIS.DIR
                    /DEBBIE.DIR
                         .
                         .
                         .
```

Figure 4. Setting Up Home Directories

USER.DIR contains all the home directories for everyone using the system. These directories should be assigned at USERDEF time;

```
USERDEF define WAYNE id 20000 DIR
/WORK1/USER.DIR/WAYNE.DIR
```

The names of the home directories should be the usernames (the name asked for at logon time) plus the suffix .DIR.

NOTE:
For normal operation, I will logon as WAYNE. When superuser priviledges are required, I can logon as SUPERWAYNE (previously defined as a secondary superuser). If you reserve logging on as SUPERUSER for the times you need to do a USERDEF, you can let the system protect you as it was designed to do.

PROJECTS

The next major directory is for our projects. Our group is working on ROBOTWELDER. Other groups are also using the NDS-II. Lump their project directories under PROJECT.DIR, too.

```
/WORK1                                  ;volume name
     /PROJECT.DIR
          /ROBOTWELDER.DIR              ;a main project
               /DATABASE.DIR
                    /SYSTEM.DIR         ;overall system database
                     Only put SVCS-type files in this directory.
                    /ASM.DIR
                     Only put SVCS-type files in this directory.
                    /DISPLAY.DIR
                     Only put SVCS-type files in this directory.
                    /database4.DIR
                     Only put SVCS type files in this directory.
                          .
                          .
                          .
          /yet--another--project.DIR
               /DATABASE.DIR
                    /SYSTEM.DIR         ;overall system database
                     Only put SVCS-type files in this directory.
                    /database.2.DIR
                     Only put SVCS-type files in this directory.
                          .
                          .
                          .
```

Figure 5. Project Directories

Put all files associated with your projects into a protected SVCS database. These file include source and objects, $INCLUDE files, MAKE files, and documents.

Break the database into many databases, each supporting a particular function or system block. In our example for the ROBOTWELDER project, one major system building block is called DISPLAY. We lump all files connected to DISPLAY into a subdirectory. DISPLAY is just one basic function of our slick new microprocessor-based ROBOTWELDER machine.

Each database should contain no more than 50-related modules to reduce database contention. The system database contains the files associated with overall project maintenance and organization. These files include block diagrams, documents and memos, timetables, and system integration test procedures.

You should not keep .LST files in a database or even on the Winchesters. They take up alot of space, and can always be regenerated when needed.

NOTE:
Only people using the network should have user names. Do not set up a user name, for example, called ROBOTWELDER.

We believe that if you do set up a project user name, such as ROBOTWELDER, with people logging on with this user name, you will lose control over your sources. It will be difficult to know who made changes on files. (Refer to the Applications Note AP 162 for further discussion.)

APPENDIX A

SYSTEM FILES

Appendix A contains a list of description of those files that the iNDX operating system maintains in the boot disk. Files beginning with r?DUP are duplicate files maintained by the operating system in case a disk "glitches." The original files are kept near the physical front of the disk, and the duplicates are kept near the back of the disk.

If the operating system detects that an original file is bad, a warning message will be printed and the duplicate files will be used. At this time, save all your files onto another device and reFORMAT the suspect disk. Otherwise, you might lose everything on your disk.

CAUTION: unless you really know what you are doing:

LEAVE ALL THE FILES LISTED BELOW ALONE!!!

DJC Queues

1. Information about DJC queues. These file can grow to contain a maximum of 256 job entries.

```
16BIT.Q          ;an import queue we created for 16-bit jobs
 8BIT.Q          ;an import queue we created for 8-bit jobs
   etc
```

HINT: Give your queues meaningful names, not like:

FOOWAFFLE or BOZO.

2. DJC header file.

```
DJC--CHK--PT     ;contains names of queues, which stations service
                 ;which queues, and the protocol version number
```

Series IV Temporary Files

1. SUBMIT jobs.

```
88--CMD--18          ;as an example
88--CMD--19          ;the naming format is 83--CMD--xy
     .
     .
     88--CMD--Y9
     etc.

     and

88--STACK--18   ;as an example
88--STACK--19
     .
     .
88--STACK--Y9
     etc
```

2. LNAMES (logical names the Series IV people are using)

```
88--LNAME--1
     .
     .
88--LNAME--J
     etc
```

Series IV and NRM CLI File

```
CLI_HELP                    ;large Series IV text file
PRM_HELP                    ;large NRM help text file

CLI_SYN_TBL                 ;used by the Series IV menu compiler
PRM_SYN_TBL                 ;used by the NRM menu compiler

HCLI                        ;Series IV error messages
PRM_HCLI                    ;NRM CLI error message
```

Series IV and NRM Logon Files

```
***Series IV
    LOGON                   ;logon CUSP
                            ;delete this and no one can logon to Series IV
    LOGON_HELP              ;online HELP text
    LOGON.SYN               ;logon menu line

***NRM
    PRM_LOGON               ;logon CUSP
                            ;delete this and no one can log on at NRM)
    PLOGON_HELP             ;online HELP text
    PLOGON.SYN              ;logon menu line
```

Electronic Mail

1. MAIL.DIR is a directory. In this directory, electronic mail will set up individual mailbox directories.

```
MAIL.DIR
```

As an example:

```
    WAYNE                   ;all of Wayne's messages will be put in this
                            ;directory
    BRIAN                   ;Brian's mailbox (MAIL uses USERDEF usernames)
```

iNDX Operating System

```
OS88.RESIDENT               ;NRM operating system, iNDX.G11 (no overlays)
OS88.OVERLAY                ;empty (length 0)
```

Communication Software

```
SYSTEM                      ;a directory that contains the following files
    CONFIG                  ;SYSGEN info (including Ethernet addresses)
    MUSER.INFO               terminal configuration info
    COMMIDOS                ;communication info
    COMM3.X02               ;Ethernet communication software
    COMM3.X03               ;Ethernet communication software
    INDX.W31                ;OS downloaded to a SeriesIV/3
    INDX.W41                ;OS downloaded to a Series IV/4
```

Disk Maintenance

```
VERIFYFIX                   ;a directory used by the VERIFY CUSP
r?BADBLOCKMAP               ;which parts of the disk not to use
r?DUPBADBLOCK               ; duplicate
r?SPACEMAP                  ;which parts of the disk are used
r?DUPSPACEMAP               ; duplicate
r?FNODEMAP                  ;which directory entries are used
r?DUPFNODEMAP               ; duplicate
r?DUPFNODE                  ;contains all file information (redundant)
                            ; Note:an iNDX disk is RMX-86-compatible
r?DUPBLOCKZERO              ;a copy of the first block on the boot disk
r?VOLUMELABEL               ;name of the disk
```

System Files

```
UDF                         ;user-definition file (NAMES + PASSWORDS)
BAK.UDF                     ;a duplicate copy you should make every time
                            ; you do a USERDEF
HOME                        ;user names and home directories
BAK.HOME                    ;a duplicate copy you should make every time
                            ; you do a USERDEF
PUBLIC.UDF                  ;public versions of UDF NAMES and home directories
BAK.PUBLIC.UDF              ;a duplicate copy you should make every time
                            ; you do a USERDEF
SPOOL                       ;the directory behind :SP:device
r?ACCOUNTING                ;not currently used
r?ISOLABEL                  ;standard ISO label
r?RESERVED1                 ;reserved for future use
r?RESERVED2                 ;reserved for future use
```

APPENDIX B

Acronyms and Definitions

iNDX (i)ntel (N)etwork (D)istributed e(X)ecutive

Intel's proprietary 16-bit operating system that runs on the NRM and the Series IV.

ISIS (i)ntel's (S)ystems (I)mplementation (S)upervisor

Intel's proprietary 8-bit operating system that runs on the Model-800, Series II, Series III, Series IV and cluster stations.

RMX (R)eal time (M)ultitasking (E)xecutive

Intel's real-time proprietary system (8-bit and 16-bit versions).

CLI (C)ommand (L)ine (I)nterpreter

CUSP (C)onnonly (U)sed (S)ystem (P)rogram

NDS-II (N)etwork (D)evelopment (S)ystem, version II

NRM (N)etwork (R)esource (M)anager

PL/M (P)rogramming (L)anguage for (M)icroprocessors

Intel's system's implementation language.

PMTs (P)rogram (M)anagement (T)ools

SVCS (S)oftware (V)ersion (C)ontrol (S)ystem, one of the PMTs

An automated means of tracking changes to program source code, maintaining variants of the source and objects modules for a program, and recording access to the source and object modules in a multiprogrammer environment.

MAKE not an acronym, one of the PMTs

A program designed to generate a submit file that can be used to construct the most current version of the requested software.